Explaining Eye Movements in Program Comprehension using jACT-R

Sebastian Lohmeier (sl@monochromata.de)
TU Berlin, Germany, M.Sc. Informatik degree course

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
TU Berlin, FG Kognitive Modellierung in dynamischen MMS
Sekr. MAR 3-2, Marchstraße 23, 10587 Berlin, Germany

Abstract

We propose that experimentally recorded sequences of eye movements are input into a cognitive model. By removing the need to model decisions on where to look next during a complex task, modelling long-term activation effects in real-world data becomes conceivable. Eye movement records from experiments on program comprehension shall be used because object-oriented source code provides knowledge structures required by a cognitive model of comprehension. We introduce a tool that supports this new approach. The tool is based on an implementation of the ACT-R cognitive architecture written in the Java programming language and could therefore attract Java developers to the cognitive modelling community.

Keywords: tools; ACT-R; Java; eye movements; program comprehension; cohesion; activation

Motivation

Eye movements of programmers reading computer programs can be explained by models written in a cognitive architecture like ACT-R (Anderson et al., 2004). This is comparable to, but may be easier to achieve, than explaining eye movements in natural language comprehension. Modelling program comprehension in a Java-based implementation of ACT-R could, in addition, attract programmers to ACT-R that are capable of contributing to ACT-R as a cognitive architecture.

Psycholinguists like Garrod and Terras (2000) have studied coherence effects (i.e. effects of semantic relations expressed in a text) in reading using eye tracking and provided explanations for their findings that are suitable for cognitive modelling. While it would be possible to model the results of these experiments in ACT-R, there are a number of open problems for such models. E.g. representations of lexical and conceptual knowledge required for these tasks are not readily available (Lohmeier & Russwinkel, 2013).

ACT-R has already been implemented in Java (see jACT-R[1] and Java ACT-R[2]) and can therefore be integrated directly into the development environments used by Java programmers, making ACT-R development possible in a well-known programming language. This would also allow ACT-R tools, e.g. for visualisation, to be developed using a larger set of existing application programming interfaces and frameworks than is available for Lisp.

Eye Movements in Program Comprehension

Mandel (1984) compared measures of eye movements during reading to predictions generated by a computational implementation of a theory from cognitive linguistics. Burkhardt, Détienne, and Wiedenbeck (1997) used a related theory to explain findings on the comprehension of the source of object-oriented computer programs. It has also been suggested that computational cognitive models be used to model program comprehension (Hansen, Lumsdaine, & Goldstone, 2012). For object-oriented programming languages provide knowledge structures required for models from cognitive linguistics (Lohmeier, 2014), eye movements of programmers are furthermore suitable for cognitive modelling based on theories from cognitive linguistics: Recorded eye movements could be used to model activation of conceptual knowledge represented in the source code. Because the source code can be parsed automatically, the conceptual knowledge expressed in the source code can be made available to the cognitive model and can be used to compute activation values to explain coherence effects in source code that are comparable to those found in studies like Garrod and Terras (2000).

Eye Tracking and jACT-R in Eclipse

We are developing a plug-in that controls an eye tracking device and sends eye movement data to a cognitive model.

Data capture and analysis

Similar to iTrace (Walters, Falcone, Shibble, & Sharif, 2013), our plug-in enables the Eclipse IDE[3] to calibrate an eye tracker, to receive data from it, to save the data to disk and to assign the data to user interface elements and words displayed at locations fixated by the user. Both iTrace and our plug-in support different eye trackers and user interface elements and can be extended to support additional ones. Our plug-in has been used to connect to eye tracking devices of SMI and The Eye Tribe and is able to map fixations to Text, StyledText, Label and Button user interface elements. Our plug-in implements complex event processing through an extensible chain of filters through which eye tracking events are passed. There are different modes, e.g. for tracking, replay and batch replay so that visualising filters can render fixations on screen during replay, but not in tracking or batch replay mode.

Interfacing eye movements and model

Preparation of data for the jACT-R model is implemented as a filter that saves saccades, fixations, and words assigned to fixations, to a log file. While tools like those of Salvucci (2000) and Heinath, Dzaack, Wiesner, and Urbas (2007) compare records of eye movements obtained experimentally to

[1] http://jact-r.org/

[2] http://cog.cs.drexel.edu/act-r/

[3] https://www.eclipse.org/

eye movements generated by a model, the model we are developing does not compute durations and targets of saccades but receives this information from the log file recorded during an experiment. The model executes cognitive processes that yield activation of knowledge representations and fixation durations. The jACT-R model is launched as a separate process within the Eclipse IDE. While data capture, analyis and submission to the model could all happen on-line, analysis and input into the model are currently separated to be able to verify the data before executing the model.

Our model does not generate saccades, but follows Salvucci (2001) in how it distinguishes but couples visual attention and the onset time of saccades, whose durations are read from the log file. For each saccade a duration is provided. Fixations come with a duration and (multiple) words within foveal and parafoveal vision. The model generates fixation durations. In addition, activation values are provided by the model. If a passage of source code has low cohesion that leads to regressive saccades in the experimental data, the model can explain this behavioural effect by means of low activation of chunks in memory that makes memory requests for a chunk referred to by a recently fixated word fail.

Discussion

The reconstructive model we aim at differs significantly from typical ACT-R models that *generate* behaviour in a *goal-directed* way. The reconstructive model attempts to execute low-level cognitive processes that fit the given fixations. There will be situations in which a sequence of fixations will end in a way that makes clear to a human analyst of the model that the model failed to execute cognitive processes that explain the behavioural data in a correct or plausible way. The better a model fits the data, the fewer of such occasions will occur. Depending on the context-dependence of the cognitive processes implemented, it may also be required to test different mutually exclusive sequences of cognitive processes and select the one that fits the behavioural data best. That would require the state of the cognitive architecture be replicated for each of such sequence and is beyond our current implementation which aims at an initial model of longer-lasting activation scenarios e.g. based on 40 minutes of eye movements.

Conclusion

While not the only possible application of ACT-R for explaining records of eye movements, eye tracking studies in program comprehension are well suited for this kind of modelling because the source code provides knowledge structures that can be used in the cognitive model. Restricting the model to low-level cognitive processes as used to establish coherence during reading should permit the implementation of a model that is able to explain rather long-term phenomena compared to models that generate goal-oriented behaviour themselves. Implementing the model in jACT-R could attact further developers to cognitive architectures.

Acknowledgments

We thank Anthony Harrison for implementing jACT-R and for helping us to get started with it.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review*, *111*(4), 1036–1060.

Burkhardt, J.-M., Détienne, F., & Wiedenbeck, S. (1997). Mental representations constructed by experts and novices in object-oriented program comprehension. In S. Howard, J. Hammond, & G. Lingaard (Eds.), *Human-computer interaction: INTERACT '97*. London: Chapman & Hall.

Garrod, S., & Terras, M. (2000). The contribution of lexical and situational knowledge to resolving discourse roles: Bonding and resolution. *Journal of Memory and Language*, *42*, 526–544.

Hansen, M. E., Lumsdaine, A., & Goldstone, R. L. (2012). Cognitive architectures: A way forward for the psychology of programming. In *Onward! 2012: Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software* (pp. 27–37).

Heinath, M., Dzaack, J., Wiesner, A., & Urbas, L. (2007). Simplifying the development and the analysis of cognitive models. In S. Vosniadou, D. Kayser, & A. Protopapas (Eds.), *Proceedings of EuroCogSci07*. Hove: Erlbaum.

Lohmeier, S. (2014). Computational linguistics vice versa. In B. du Boulay & J. Good (Eds.), *Proceedings of the psychology of programming interest group annual conference 2014* (pp. 191–196).

Lohmeier, S., & Russwinkel, N. (2013). Issues in implementing three-level semantics with ACT-R. In *Proceedings of the 12th international conference on cognitive modeling (ICCM)*.

Mandel, T. S. (1984). *An eye movement investigation of a process model of comprehension* (Tech. Rep. No. 84-132). University of Colorado, Institute of Cognitive Science.

Salvucci, D. D. (2000). An interactive model-based environment for eye-movement protocol analysis and visualization. In *Proceedings of the 2000 symposium on eye tracking research & applications* (pp. 57–63). New York: ACM.

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, *1*(4), 201–220.

Walters, B., Falcone, M., Shibble, A., & Sharif, B. (2013). Towards an eye-tracking enabled IDE for software traceability tasks. In *International workshop on traceability in emerging forms of software engineering (TEFSE), 2013* (pp. 51–54).